

PCTWORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : G06F 17/00	A2	(11) International Publication Number: WO 98/47080 (43) International Publication Date: 22 October 1998 (22.10.98)
<p>(21) International Application Number: PCT/US98/07660</p> <p>(22) International Filing Date: 15 April 1998 (15.04.98)</p> <p>(30) Priority Data: 08/838,082 15 April 1997 (15.04.97) US</p> <p>(71) Applicant: ION, INC. [US/US]; 216 San Francisco Boulevard, San Anselmo, CA 94960 (US).</p> <p>(72) Inventors: ROBERTS, Dale, Tyson; 164 Spencer Avenue, Sausalito, CA 94965 (US). GREENBERG, Ann, E.; 216 San Francisco Boulevard, San Anselmo, CA 94560 (US).</p> <p>(74) Agents: BACKOFEN, Paul, J. et al.; Irell & Manella LLP, Suite 900, 1800 Avenue of the Stars, Los Angeles, CA 90067 (US).</p>		<p>(81) Designated States: AU, CA, JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).</p> <p>Published <i>Without international search report and to be republished upon receipt of that report.</i></p>
<p>(54) Title: NETWORK DELIVERY OF INTERACTIVE ENTERTAINMENT COMPLEMENTING AUDIO RECORDING</p> <p>(57) Abstract</p> <p>Entertainment content complementary to a musical recording is delivered to a user's computer by means of a computer network link. The user employs a browser to access the computer network. A plug-in for the browser is able to control an audio CD or other device for playing the musical recording. A script stored on the remote computer accessed over the network is downloaded. The script synchronizes the delivery of the complementary entertainment content with the play of the musical recording.</p>		

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakhstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

NETWORK DELIVERY OF
INTERACTIVE ENTERTAINMENT
COMPLEMENTING AUDIO RECORDINGS

BACKGROUND

5 1. Field of the Invention

 This invention pertains to the field of computer networking, and more particularly to the use of network protocols to provide services to users that are related to CD ROMs, audio recordings and other distributed media.

10 2. Related Art

 Over the past few years, on-line services have experienced explosive growth and have become a major new form of entertainment. Alongside this new entertainment, more traditional forms such as musical recordings have continued to
15 be consumed on a massive scale.

 The traditional experience of the musical recording is listening by a small group of persons gathered together in a room. The music fills the room acoustically, but there is little associated visual content, and there is only a limited
20 interaction with the recording, consisting essentially of deciding which tracks to play and performing simple transformations on the recorded sound, such as setting the volume or applying an audio equalizer. This traditional experience dates back to the early age of 78 r.p.m. musical
25 recordings almost a century ago.

 The traditional production of a musical recording complements the traditional experience of the recording. The recording is produced in a number of recording sessions, subject to careful mixing and editing, and then released to the public.

5 Music videos have supplemented the traditional experience
of musical recordings by allowing the association of visual
content with tracks of such a recording. In practice, however,
music videos have been broadcast, with all the problems of lack
of user control which that implies, and they have not
10 contributed to interactivity or participation by the consumer.

15 SUMMARY OF THE INVENTION

It is therefore an object of this invention to provide computer programs, systems, and protocols which allow producers to deliver entertainment complementary to distributed media recordings by means of on-line services such as the Internet.

20 It is a further object of this invention to provide computer programs, systems, and protocols which allow such complementary entertainment to be meaningfully interactive for the consumer, such that the consumer can also be a creator of the experience.

It is a further object of the invention to achieve the foregoing objects by means of implementations designed to attain integration with existing environments and programs, particularly on the Internet, while retaining the flexibility to adapt to the continuing evolution of standards for on-line services.

In a first aspect of the present invention provides a means for producers and sellers of distributed media such as CDs to maintain and strengthen their connection to their customers. Record companies download and periodically update a central
5 library of complimentary content for CD's the company has in the market. The software of the present invention operate as a plug-in to a users web browser and directs a user with a record company's CD to a particular section of the central library appropriate for the user's CD.

10 In another aspect of the present invention, called "CD Watcher" data representative of the users listening habits relative to a record company's CD is transferred to the record company when complimentary content is delivered to the user over a network connection.

15 Alternatively, record companies contributing and maintaining the central library have access to the listening habits of all users who have accessed the central library for complimentary content.

In another aspect of the invention, software is provided
20 which permits a computer program running on a remote host to control a distributed media player such as a compact disk (CD) player, DVD player, or the like on a user's computer. (For convenience, we use the term "CD" to refer to all distributed media and the term "CD player" to refer also to all distributed
25 media players such as DVD players and similar devices.) The software is designed to permit the remote host both to initiate actions on the CD player and to become aware of actions which the user has initiated by other control means, such as the buttons on the CD player's front panel or a different CD player
30 control program. This aspect of the invention is a building-block for the provision of complementary entertainment for CD

content when those recordings are fixed in the prevailing contemporary form, the CD.

In another aspect of the invention, visual content, including interactive content, may be delivered over an on-line service in such a way that it is synchronized to the delivery of content from a musical recording. Such visual content may, for example, be synchronized to the playing of an audio CD or other distributed media in the user's computer. The visual content is thematically linked to the musical recording, for example in the manner of a music video.

In a further aspect of the invention, a method is provided for determining or assigning a substantially unique identifier to CD or other distributed media content consisting of a number of tracks. A unique identifier is a useful complement to the delivery of supplementary content in conjunction with the playing of a CD or other distributed media in that it allows the software which delivers the supplementary content to be sure that the CD is in fact the correct CD to which the supplementary content corresponds. If the supplementary content is designed, for example, to accompany the Rosary Sonatas of Heinrich Ignaz Franz Biber, it would presumably not function well if the CD or other distributed media in the user's player were the soundtrack for the film Mary Poppins. The unique identifier also allows a CD or other distributed media to be used as a key to access a premium Web area. Furthermore, the unique identifier can allow the user to be directed to an area of the Web corresponding to the CD or other distributed media which is in the user's machine.

In a still further aspect of the invention, the immensely popular on-line service generally referred to as a "chat room" may be enhanced by means of a link to a CD recording which all persons in the room are playing. A remote host may control

distributed media players in multiple remote locations. The chat room experience as it exists today in on-line services has a disembodied quality by comparison with traditional face-to-face social encounters in which there are identifiable surroundings.

5 The only common experience to the chat users today are the words of the chat as they fly by on a computer screen, and perhaps the user icons ("avatars") or other visual content occupying a small space on the screen. The use of a musical recording in conjunction with a chat room opens up the possibility of

10 restoring to the experience a degree of the shared ambience of traditional social encounters. Furthermore, shared content such as a musical recording offers a focal point that allows chat-seekers to group together by means of shared interests in a particular type of recording.

15

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a network diagram of a first embodiment of the present invention.

Fig. 2 is a flow diagram of the embodiment shown in Fig. 1.

Fig. 3 is a block diagram of the environment in which the

20 present invention operates.

Fig. 4 is a block diagram of complimentary content according to an aspect of the present invention.

Fig. 5 is a flowchart of the synchronization code of the invention.

25 Fig. 6 is a flowchart of the sequence of operations for connecting the present invention to a chat room.

DESCRIPTION OF PREFERRED EMBODIMENTS

Referring to Fig.'s 1 and 2, an embodiment of the present invention dynamically connects a user playing a CD with a remote host for data exchange. At block 11, the user links to a remote host and requests a download of client 26. To receive the download the user must provide some basic information, name, e-mail, chat name, etc. The registration information is exchanged for client 26 at block 11A. At block 10A user computer 10 is running computer program 12 such as a browser with client 26. At block 30A insertion of CD 30 into player 32 triggers action by client 26. Client 26 takes control of player 32 and scans CD 30. CD 31 may include time code 31A and other encoded data 31B. Client 26 uses the results of the CD scan to calculate a substantially unique CD ID 31, block 34. Some CD's may contain an ID text file or IRC code, there is no universal standard, thus an ID calculation technique may yield a useful ID with any CD. This technique may create a pattern match 31F by sampling a subset of the content of CD 30 and using the sample to create a substantially unique pattern match 31F of CD 30. While client 26 is calculating CD ID 31, at block 36 use demographic data 62 is also collected and temporarily stored by client 26.

Use demographic data 62 includes but is not limited to CD use profiles including most used tracks, total time of use, most used CDs, average length of time computer 10 is running, software loaded, most used software, software running concurrently with client and the like.

Calculation of a CD ID 31 stimulates client 26 to direct computer program 12 to check a local cache for CD ID 31. IF CD ID 31 is not present in the local cache client 26 links to look-up server 40 at block 38. If computer program 12 is not already running, client 26 may launch it. Once the link to look-up server 40 is established, client 26 sends CD ID 31 to look-up

server 40. Look-up server 40 compares CD ID 31 against table 42. Table 42 is a look-up table linking CD ID's with some associated content and with particular addresses having complimentary content. Other information such as timing and control data, electronic coupons, advertisements and bonus content such as video with timing and control data may also be contained in table 42. At block 44, look-up server 40 sends information such as address 46 to the user in response to receipt of CD ID 31. The information sent to the user may or may not be based on the user demographic data and use demographic data 62 sent to look-up server 40. At block 48 client 26 establishes a link to address 46. Address 46 may be a premium or subscription site such as site 51A in which case CD ID 31 may operate as a password.

15 In one aspect of the present invention, content suppliers 51-56 such as record companies 1-6 respectively, maintain a central library 50 on the web. Content suppliers may also include advertisers, CD retailers, and other content rights holders. Central library 50 may be on a single server such as look-up server 40 or it may be distributed. Central library 50 contains the complimentary content sites such as site 51A linked by the addresses in table 42 such as address 46. Each content supplier 51-56 may change the content of their site and add new links to additional sites as new CD's are released. As new sites are added, new CD ID's and linked addresses are added to table 42.

At block 49A, server 58 hosting site 51A transfers complimentary content 60 to the user's computer 10. Once the user is linked to site 51A, client 26 sends stored use demographic data 62 and CD status data 64 to look-up server 40 and or server 58. At block 66 client 26 continues to update and transfer use demographic data 62 and CD status data 64 to look-

up server 40 and or server 58 as long as client 26 is running. Closed loop update 70 permits a content supplier, such as content suppliers 51-56, a real-time or near real-time look at which CDs are in use and relative frequency of use of CD
5 elements and related information. Closed loop update 70 also provides control of distributed media playing on a users computer 10. A removal or change of CD at block 68 would return the client to block 30A.

In another embodiment, this invention operates on the World
10 Wide Web. The HTTP protocol on the web is run atop a general connection-oriented protocol, which today is generally TCP/IP, described in Douglas E. Comer, *Internetworking with TCP/IP* (3d ed. 1995). However, the invention described here is not limited to HTTP running over any particular kind of network software or
15 hardware. The principles of the invention apply to other protocols for access to remote information that may come to compete with or supplant HTTP.

Referring now to Fig. 3, a user sits at his or her computer
10 and runs a computer program 12 such as a browser or other client software. The browser sends out HTTP requests 14 to
20 other computers, referred to as servers such as server 16. In requests, particular items of data, referred to as resources, which are available on servers, are referred to by means of uniform resource locators (URL's), character strings in a
25 particular format defined in Berners-Lee et al., supra. A URL includes both an identification of the server and an identification of a particular item of data within the server. Reacting to the requests, the servers return responses 18 to the user's browser, and the browser acts upon those responses,
30 generally by displaying some sort of content to the user.

The content portion of the responses can be a "Web page," expressed in the hypertext markup language (HTML) such as pages

20 and 22. That language allows one to express content consisting of text interspersed with bitmap-format images and links (also known as anchors and hyperlinks). The links are further URL's to which the browser may, at the user's prompting,
5 send further requests.

The responses can also include more complex commands to be interpreted by the browser, e.g., commands which result in an animation as discussed below for Fig. 4. HTML itself does not define complex commands, but rather they are considered to
10 belong to separately-defined scripting languages, of which two currently common ones are JavaScript and VBScript.

In addition to extending the function of the browser by means of code written in a scripting language, it is also possible to extend the function of a browser with compiled code.
15 Such compiled code is referred to as a "plug-in." The precise protocol for writing a plug-in is dependent on the particular browser. Plug-ins for the Microsoft browser are referred to by the name of ActiveX controls.

Plug-ins may be very complex. A plug-in which may
20 advantageously be used in connection with the invention is Shockwave from Macromedia. It permits animations which are part of a server response to be downloaded and played to the user. Shockwave defines its own scripting language called Lingo. Lingo scripts are contained within the downloadable animations
25 which the Shockwave plug-in can play. The general format of a Shockwave animation is a timeline consisting of a series of frames, together with a number of visual objects which appear, perform motions, and disappear at particular frames within the timeline. To achieve more complex effects within a Shockwave
30 animation, Lingo scripts may be invoked in addition to predefined visual objects.

A currently preferred embodiment of the present invention employs a plug-in, referred to as the command plug-in 24, which provides to a scripting language the ability to command and monitor in a detailed fashion the playing of a CD recording.

5 The command plug-in should provide and monitor, at a minimum, the following basic functions:

- (1) Start and stop play.
- (2) Get current track and position within the track.
- (3) Seek to a track and a position within the track.
- 10 (4) Get and set volume.
- (5) Get information regarding the CD (e.g., the number of tracks, their lengths, the pauses between tracks).
- (6) Get information regarding the capabilities of the CD drive.

15 Other functions may be provided and monitored, limited only by what the underlying operating system services are able to accommodate. The monitored functions are included in use demographic data 62 which is transferred to server 40 and or servers such as server 58.

20 The command plug-in may be written in a conventional programming language such as C++. The plug in must conform to the existing standards for plug-ins, such as those required of Microsoft ActiveX objects. In order to obtain the information and carry out the functions which the command plug-in makes
25 available to the scripting language, the command plug-in relies on functions which provide control and information regarding the playing musical recording. These functions will depend on the precise source of the recording. If, as in one embodiment of the present invention, the recording being played is an audio CD
30 in the computer CD player, and if the browser is running under Microsoft Windows 3.1 or Windows 95 or Windows CE, these functions would be the MCI functions, which form a part of the

Win32 application programming interface. These functions are documented, for example, in Microsoft Win32 Programmer's Reference. Different functions may be provided by streaming audio receivers, as for example receivers which capture audio
5 which is coming into the user's computer over a network connection in a suitable audio encoding format such as MPEG.

An important point to note about the implementation of the command plug-in is that the operations which it carries out, as for example "seeks", may take times on the order of a second.
10 It is undesirable for command-plug-in 24 to retain control of computer 10 during that interval, so it is important that command plug-in 24 relinquish control of computer 10 to the browser whenever a lengthy operation is undertaken, and report on the results of the operation via the asynchronous event
15 handling capability used in the common scripting languages.

Given the above summary of the functions which the command plug-in provides, a general knowledge of how to write plug-ins (e.g., of how to write ActiveX objects), and a knowledge of the relevant application programming interface for controlling the
20 play of the CD (e.g., MCI in Win32), a person skilled in the art could readily and without undue experimentation develop an actual working command plug-in. For this reason, further details of how the command plug-in is implemented are not provided here.

25 The existence of a command plug-in providing the functions listed above to a scripting language is a foundation on which entertainment complementary to the content of a CD may be constructed. In particular, it is possible to devise, building on this foundation, a method for synchronizing the display of
30 complementary content by means of the scripting language with the events which are occurring on the CD.

II

Referring now to Fig.'s 4 and 5, synchronization of the complementary content to the CD proceeds as follows. For example, complimentary content 60 may be provided by means of animation such as Shockwave animation, including frames F_1-F_n and script 72. Complimentary content 60 is downloaded from server 58 and displayed for the user by means of a Shockwave plug-in. This downloading may take place before the animation is displayed, or alternatively it make take place as the animation is being displayed, provided the user's connection to the network is fast enough to support download at an appropriate speed. The downloading is a function provided by the Shockwave plug-in itself.

As the Shockwave animation is played, script 72 such as a Lingo script executes each time a frame F_n finishes displaying. The Lingo script contains a description of the relationship which should exist between frames of the animation and segments of the CD content, identified by track number and by time. The Lingo script determines, by means of the command plug-in described above, at which track and time the play of the CD is. It then refers to the description in order to determine which frames of the animation correspond to that portion of the CD. If the current frame is not one of those frames, the Lingo script resets the time line of the animation so that the animation will begin to play at the frame which corresponds to the current position of the CD. This permits the visual content to catch up if it ever lags the CD, for example because downloading from the network has fallen behind, because the user's computer lacks the cycles to play the animation at full speed, or because the user has fast-forwarded the CD.

Referring now to Fig 4, the synchronization algorithm may control individual frames or groups of contiguous frames. complimentary content 60 includes frames F_1-F_n and script 72. At

block 200, a correspondence is established between each frame F_n or group of frames and a particular segment of CD 30. At the end of each frame F_n of the animation block 205, the position of CD 30 is determined, block 210. A test is done at block 215 to
5 determine whether the position of CD 30 is within the segment of the recording that corresponds to the group of frames to which the next sequential frame belongs. If the position of CD 30 is within that segment, the playback of the animation proceeds with the next frame, block 230. If the position of CD 30 is not
10 within that segment, then at blocks 220 and 225 the playback of the animation is advanced to the frame corresponding to where the CD is.

A further aspect of the present invention is a touring mode. With client 26 in touring mode, a remote device such as
15 server 58 may control one or more user devices through delivery of complimentary content 60 interspersed with player 26 control data to provide the one or more users with a guided tour of a particular distributed media such as CD 30.

A further aspect of the invention is the ability, by making
20 use of command plug-in 24, to provide a technique for establishing a unique identifier for a CD, CD ID 31, which is located in the user's CD player 32. The unique identifier may be based on the number and lengths of the tracks (measured in blocks, i.e., 1/75ths of a second), so that the identifier would
25 be a concatenation of these lengths. In practice, however, it is desirable to have a somewhat shorter identifier, so the unique identifier is preferably the concatenation of the track lengths expressed in a fairly coarse unit, such as 1/4th of a second.

30 Appendix A contains source code, written in C, for a fuzzy comparison algorithm suitable for determining whether two audio CDs are exactly or approximately the same. The fuzzy comparison

algorithm proceeds as follows. For each of the two audio CDs to be compared, one determines the lengths of all the tracks in the recordings in milliseconds. One then shifts all track lengths to the right by eight bits, in effect performing a truncating
5 division by $28 = 256$. One then goes through both of the recordings track by track, accumulating as one proceeds two numbers, the match total and the match error. These numbers are both initialized to zero at the start of the comparison. For each of the tracks, one increments the match total by the
10 shifted length of that track in the first CD to be compared, and one increments the match error by the absolute value of the difference between the shifted lengths of the track in the two CDs. When one gets to the last track in the CD with the fewer number of tracks, one continues with the tracks in the other CD,
15 incrementing both the match total and the match error by the shifted lengths of those tracks. Following these steps of going through the tracks, the algorithm then divides the match error by the match number, subtracts the resulting quotient from 1, and converts the difference to a percentage which is indicative
20 of how well the two CDs match.

Appendix B contains source code, written in C, for a comparison algorithm suitable for determining whether two audio CDs are exactly the same. The algorithm generates from the number of tracks, the track lengths, and the start and end times
25 of the tracks an 8-byte value. The high order 4 bytes are obtained by summing the start and end times of all tracks, expressed in milliseconds. The low order 4 bytes are obtained by summing the lengths of all tracks expressed in milliseconds, shifting the sum left ten bits, and adding the number of tracks.

30 CD ID 31 may be employed as a database key. A site such as site 52A may maintain a database of information about CDs, for example information about all CDs issued by record company 2 can

be maintained on that record company's site. There are various alternative ways for users to navigate this information. For example, they could use a Web page containing many hyperlinks as a table of contents, or they could use a conventional search engine. A third way of searching, which is enabled by CD ID 31 of the invention, is for there to be a Web page which invites the user to place in player 32 the CD about which he or she is seeking information, for example CD 30. Upon detection of the presence of CD 30 in the drive, a script in the Web page computes CD ID 31 corresponding to CD 30 and sends it to server 58. Server 58 then displays information about the CD retrieved from a database on the basis of CD ID 31. This information may include a Web address (URL) that is related to the CD (e.g., that of the artists' home page), simple data such as the names of songs on the CD, and also complementary entertainment, including potentially photographs (e.g., of the band), artwork, animations, and video clips. It is also possible to arrange things so that, when the user inserts a CD into the computer, (i) the Web browser is launched if not already running, (ii) the browser computes the CD's unique identifier and from that unique identifier derives a URL, and (iii) the browser does an HTTP get transaction on that URL.

An alternative application of unique identifiers for musical recordings is to employ a CD as a key for entering into a premium area of the Web. There are presently premium areas of the Web to which people are admitted by subscription. A simple form of admission based on the unique identifier is to require, before accessing a particular area of the Web, that the user place in his or her CD drive a particular CD, or a CD published by a particular company or containing the music of a particular band or artist. This is readily accomplished by means of a script which invokes the functions provided by the command plug-in and computes a unique identifier.

Another aspect of the invention is the connection of chat rooms with musical recordings. The goal is to provide all participants in a chat room with the same music at approximately the same time.

5 One conventional network protocol for chat services is Interney Relay Chat (IRC), described J. Oikarinen & D. Reed, Internet Relay Chat Protocol (Internet Request for Comments No. 1459, 1993). In this protocol, when one becomes a client of a chat server, one sends the name of a chat room. The chat server
10 receives messages from all of its of clients and relays the messages sent in by one client to all the other clients connected in the same room as that client. The messages which a client sends are typically typed in by the user who is running the client, and the messages which a client receives are
15 typically displayed for the user who is running the client to read.

In a preferred embodiment of the invention, a chat client is customized by means of a plug-in, which we will call the chat plug-in. The chat client is started up by a browser as follows
20 (see Fig. 5). The user connects by means of the browser to a central Web page (box 300) which, upon being downloaded, asks that the user insert a CD into his or her player (box 305). A unique identifier of the CD is computed and communicated back to the server by using the control plug-in described above under
25 the command of a script in the central Web page (box 310). The server then employs the unique identifier to determine whether it has a chat room focused on the CD (box 315). This step may be carried out by looking the unique identifier up in a database using techniques well known in the art. There exists a vast
30 literature on connecting Web pages to databases, e.g., December & Ginsburg, supra, chapter 21. If a chat room focused on the CD exists or can be created, the server responds with the name of

that chat room, and the browser starts up a chat client on the user's computer as a client of that chat room (box 320).

The chat room's name is set by the server to contain information about the track which the CD is playing in the other
5 chat room clients' machines and the time at which the track started to play, as well as about the volume at which the CD is playing. The chat client plug-in employs that information to direct the control plug-in to set the CD in the user's computer to play in such a manner that it is approximately synchronized
10 to the CD which is playing in the other chat room clients' machines (box 320).

Each user in the chat room is able to control the CD which is playing in his or her machine. Control actions result in the chat plug-in sending messages to the chat server which describe
15 the control action being taken (box 325). For example, such messages may indicate a change in the position of the CD, a change in the volume, or the ejection of the CD to replace it with another. The chat plug-ins running on the other users' machines, upon seeing a message of this kind, replicate the
20 action (as far as possible) on the other users' machines by using the control plug-in described above (box 330).

In a further aspect of the invention, a chat room focused on a particular musical recording might allow for a voting procedure to select particular tracks. A simple voting
25 procedure would be for each chat plug-in to act upon a change message of the kind described in the preceding paragraph only when it sees two identical consecutive change messages. This would mean that in order to change the track which is being played, it would be necessary for two users to change to that
30 track. The number two may be replaced by a higher number.

In a further aspect of the invention the messages delivered to the users of a chat can be driven from a text file rather than manual typing. This would allow a pre-recorded experience to be played back for a group of chat users. Such a technique
5 may be used to create a pre-recorded, narrated tour of an audio CD.

An important advantage of the embodiment described above is that it may be used with any chat server software which supports the minimal functionality required by Internet Relay Chat or by
10 a protocol providing similar minimum chat service. The additional software required is located in the chat client plug-in and in the central Web page, with its connection to a database of CD information.

APPENDIX A

```

/*
 * FUZZY CD ID
 * (c) 1996-1998 ION Inc.
 *
 * by Ty Roberts
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

struct fuzzyCDid {
    short          numberTracks; // start time in
milliseconds
    unsigned short fuzzlength[100];
};

typedef struct fuzzyCDid fuzzyCDid, *fuzzyCDidPtr;

// structure of a cd track with all times stored in milliseconds

struct cdtrack {
    long beginMs; // start time in milliseconds
    long endMs;   // end time in milliseconds
    long lengthMs; // length in milliseconds
};

typedef struct cdtrack cdtrack, *cdTrackPtr;

struct cd {
    short    numberTracks;
    cdtrack  track[100];
};

typedef struct cd cd, *cdPtr;

void CreateFuzzyId( fuzzyCDidPtr fid, cdPtr cd );

float FuzzyMatch( fuzzyCDidPtr fid1, fuzzyCDidPtr fid2 );

// SUBROUTINES

void CreateFuzzyId( fuzzyCDidPtr fid, cdPtr cd )
{

```

```

    long    i;

    // first copy in the number of tracks
    fid->numberTracks = cd->numberTracks;

    for(i=0;i<fid->numberTracks;i++) {
        // shift left and create a MSB length thats not exact
        fid->fuzzlength[i] = (short)(cd-
>track[i].lengthMs>>8);
    }
}

float      FuzzyMatch( fuzzyCDidPtr fid1, fuzzyCDidPtr fid2 )
{
    long      fidmatcherr = 0, fidmatchtotal = 0;
    short      i, trackcnt;
    float      matchpercent;

    // find the larger number of tracks
    trackcnt = fid1->numberTracks<fid2->numberTracks ? fid2-
>numberTracks : fid1->numberTracks;

    // cycle thru the tracks accumulating error and total
    comparedtimes
    for(i=0;i<trackcnt;i++) {
        if ((i < fid1->numberTracks) && (i < fid2-
>numberTracks)) {
            fidmatcherr += abs(fid1->fuzzlength[i] - fid2-
>fuzzlength[i]);
            fidmatchtotal += fid1->fuzzlength[i];
        } else if (i >= fid2->numberTracks) {
            fidmatcherr += fid1->fuzzlength[i];
            fidmatchtotal += fid1->fuzzlength[i];
        } else if (i >= fid1->numberTracks) {
            fidmatcherr += fid2->fuzzlength[i];
            fidmatchtotal += fid2->fuzzlength[i];
        }
    }

    if (fidmatcherr > 0) {
        matchpercent = 100 -
(((float)fidmatcherr/((float)fidmatchtotal) *100);
    } else {
        matchpercent = 100;
    }
    return matchpercent;
}

```

```

void main(void)
{
    short i;
    float    matchpercent;

    // create global structures for two complete cds with up to
100 tracks
    cd    cd2id;
    fuzzyCDid fidcd2id;
    cd    cdFromDB;

    fuzzyCDid fidcdFromDB;

    printf ("Test #1 will compare two CDs that are exactly the
same\n\n");

    // put in some test values for the cd track lengths
    // since these are in ms, its basically 60000 = 1 minute

    cd2id.track[0].lengthMs = 121323;
    cd2id.track[1].lengthMs = 234565;
    cd2id.track[2].lengthMs = 566437;
    cd2id.track[3].lengthMs = 245120;
    cd2id.track[4].lengthMs = 20000;
    cd2id.track[5].lengthMs = 120386;
    cd2id.track[6].lengthMs = 323453;
    cd2id.numberTracks = 7;

    for(i=1;i<cd2id.numberTracks;i++) {
        printf ("CD #1: Track = %d  length in minutes = %f\n",
            i, (float)cd2id.track[i].lengthMs/60000.0);
    }
    printf("\n");

    cdFromDB.track[0].lengthMs = 121323;
    cdFromDB.track[1].lengthMs = 234565;
    cdFromDB.track[2].lengthMs = 566437;
    cdFromDB.track[3].lengthMs = 245120;
    cdFromDB.track[4].lengthMs = 20000;
    cdFromDB.track[5].lengthMs = 120386;
    cdFromDB.track[6].lengthMs = 323453;
    cdFromDB.numberTracks = 7;

    for(i=1;i<cdFromDB.numberTracks;i++) {
        printf ("CD #2: Track = %d  length in minutes = %f\n",

```

```

        i, (float)cdFromDB.track[i].lengthMs/60000.0 );
    }

    CreateFuzzyId( &fidcd2id, &cd2id );
    CreateFuzzyId( &fidcdFromDB, &cdFromDB );

    matchpercent = FuzzyMatch( &fidcd2id, &fidcdFromDB );
    printf ("The cd's matchpercent was computed as=%f",
matchpercent);
    printf ("\n");
    printf ("\n");

    printf ("Test #2 will compare two cd that are nearly the
same\nexcept they have diffent # of tracks \n");

    // put in some test values for the cd track lengths
    // since these are in ms, its basically 60000 = 1 minute
    cd2id.track[0].lengthMs = 121323;
    cd2id.track[1].lengthMs = 234565;
    cd2id.track[2].lengthMs = 566437;
    cd2id.track[3].lengthMs = 245120;
    cd2id.track[4].lengthMs = 20000;
    cd2id.track[5].lengthMs = 120386;

    cd2id.track[6].lengthMs = 323453;
    cd2id.numberTracks = 7;

    for(i=1;i<cd2id.numberTracks;i++) {
        printf ("CD #1: Track = %d length in minutes = %f\n",
            i, (float)cd2id.track[i].lengthMs/60000.0 );
    }
    printf ("\n");
    cdFromDB.track[0].lengthMs = 121323;
    cdFromDB.track[1].lengthMs = 234565;
    cdFromDB.track[2].lengthMs = 566437;
    cdFromDB.track[3].lengthMs = 245120;
    cdFromDB.track[4].lengthMs = 20000;
    cdFromDB.track[5].lengthMs = 120386;
    cdFromDB.numberTracks = 6;

    for(i=1;i<cdFromDB.numberTracks;i++) {
        printf ("CD #2: Track = %d length in minutes = %f\n",
            i, (float)cdFromDB.track[i].lengthMs/60000.0 );
    }

    CreateFuzzyId( &fidcd2id, &cd2id );

```



```

        CreateFuzzyId( &fidcdFromDB, &cdFromDB );
        matchpercent = FuzzyMatch( &fidcd2id, &fidcdFromDB );

        printf ("The cd's matchpercent was computed
as=%f",matchpercent);
        printf ("\n");
        printf ("\n");
        printf ("Test #3 will compare two cd that are not the
same\n\n");

        // put in some test values for the cd track lengths
        // since these are in ms, its basically 60000 = 1 minute
        cd2id.track[0].lengthMs = 34213;
        cd2id.track[1].lengthMs = 334565;
        cd2id.track[2].lengthMs = 231423;
        cd2id.track[3].lengthMs = 134122;
        cd2id.track[4].lengthMs = 2342;
        cd2id.track[5].lengthMs = 3487;
        cd2id.track[6].lengthMs = 9976;
        cd2id.numberTracks = 7;

        for(i=1;i<cd2id.numberTracks;i++) {
            printf ("CD #1: Track = %d  length in minutes = %f\n",
                i, (float)cd2id.track[i].lengthMs/60000.0 );
        }
        printf ("\n");
        cdFromDB.track[0].lengthMs = 121323;
        cdFromDB.track[1].lengthMs = 234565;
        cdFromDB.track[2].lengthMs = 566437;
        cdFromDB.track[3].lengthMs = 245120;
        cdFromDB.track[4].lengthMs = 20000;
        cdFromDB.track[5].lengthMs = 120386;
        cdFromDB.track[6].lengthMs = 323453;
        cdFromDB.numberTracks = 6;

        for(i=1;i<cdFromDB.numberTracks;i++) {
            printf ("CD #2: Track = %d  length in minutes = %f\n",
                i, (float)cdFromDB.track[i].lengthMs/60000.0 );
        }

        CreateFuzzyId( &fidcd2id, &cd2id);
        CreateFuzzyId( &fidcdFromDB, &cdFromDB);

        matchpercent = FuzzyMatch( &fidcd2id, &fidcdFromDB );

```

```
        printf ("The cd's matchpercent was computed  
as=%f",matchpercent);  
    }
```

APPENDIX B

```
/*
 * EXACT MATCH CD ID
 * © 1996-1998 ION Inc.
 *
 * by Ty Roberts
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

struct cdid{
    long id[2];
};

typedef struct cdid cdid, *cdidPtr;

// structure of a cd track with all times stored in milliseconds

struct cdtrack{
    long beginMs; // start time in milliseconds
    long endMs;   // end time in milliseconds
    long lengthMs; //length in Milliseconds
};

typedef struct cdtrack cdtrack, *cdTrackPtr;

struct cd {
    short    numberTracks;
    cdtrack  track[100];
};

typedef struct cd cd, *cdPtr;

void CreateUniqueId( cdidPtr cid, cdPtr cd );

// SUBROUTINES
void CreateUniqueId( cdidPtr cid, cdPtr cd )
{
    long    i, t, n;

    t = 0;
    n = 0;
```

```

        for(i=0;i<cd->numberTracks;i++) {
            // shift left and create a MSB length thats not exact
            t += cd->track[i].lengthMs;
            n += cd->track[i].beginMs + cd->track[i].endMs;
        }
        cid->id[0] = t<<10+cd->numberTracks;
        cid->id[1] = n;
    }

void main(void)
{
    short i;
    short      matchtest;

    // create global structures for two complete cds with up to
100 tracks
    cd  cd2id;
    cdid cd2UID;

    cd  cdFromDB;
    cdid cdFromDBUID;

    printf ("Test #1 will compare two cd that are exactly the
same\n\n");

    // put in some test values for the cd track lengths
    // since thes are in ms, its basically 60000 = 1 minute
    cd2id.track[0].beginMs = 0;
    cd2id.track[1].beginMs = 100001;
    cd2id.track[2].beginMs = 231001;
    cd2id.track[3].beginMs = 345001;
    cd2id.track[4].beginMs = 435001;
    cd2id.track[5].beginMs = 460001;
    cd2id.track[6].beginMs = 590001;

    cd2id.track[0].endMs = 100000;
    cd2id.track[1].endMs = 231000;
    cd2id.track[2].endMs = 345000;
    cd2id.track[3].endMs = 435000;
    cd2id.track[4].endMs = 460000;
    cd2id.track[5].endMs = 590000;
    cd2id.track[6].endMs = 690000;
    cd2id.track[0].lengthMs = cd2id.track[0].endMs -
cd2id.track[0].beginMs;

```

2

```

        cd2id.track[1].lengthMs = cd2id.track[1].endMs -
cd2id.track[1].beginMs;
        cd2id.track[2].lengthMs = cd2id.track[2].endMs -
cd2id.track[2].beginMs;
        cd2id.track[3].lengthMs = cd2id.track[3].endMs -
cd2id.track[3].beginMs;
        cd2id.track[4].lengthMs = cd2id.track[4].endMs -
cd2id.track[4].beginMs;
        cd2id.track[5].lengthMs = cd2id.track[5].endMs -
cd2id.track[5].beginMs;
        cd2id.track[6].lengthMs = cd2id.track[6].endMs -
cd2id.track[6].beginMs;
        cd2id.numberTracks = 7;

        for(i=1;i<cd2id.numberTracks;i++) {
            printf ("CD #1: Track = %d    length inminutes = %f\n",
i, (float)cd2id.track[i].lengthMs/60000.0 );
        }
        printf ("\n");
        cdFromDB.track[0].beginMs = 0;
        cdFromDB.track[1].beginMs = 100001;
        cdFromDB.track[2].beginMs = 231001;
        cdFromDB.track[3].beginMs = 345001;
        cdFromDB.track[4].beginMs = 435001;
        cdFromDB.track[5].beginMs = 460001;
        cdFromDB.track[6].beginMs = 590001;
        cdFromDB.track[0].endMs = 100000;
        cdFromDB.track[1].endMs = 231000;
        cdFromDB.track[2].endMs = 345000;
        cdFromDB.track[3].endMs = 435000;
        cdFromDB.track[4].endMs = 460000;
        cdFromDB.track[5].endMs = 590000;
        cdFromDB.track[6].endMs = 690000;
        cdFromDB.track[0].lengthMs = cd2id.track[0].endMs -
cd2id.track[0].beginMs;
        cdFromDB.track[1].lengthMs = cd2id.track[1].endMs -
cd2id.track[1].beginMs;
        cdFromDB.track[2].lengthMs = cd2id.track[2].endMs -
cd2id.track[2].beginMs;
        cdFromDB.track[3].lengthMs = cd2id.track[3].endMs -
cd2id.track[3].beginMs;
        cdFromDB.track[4].lengthMs = cd2id.track[4].endMs -
cd2id.track[4].beginMs;
        cdFromDB.track[5].lengthMs = cd2id.track[5].endMs -
cd2id.track[5].beginMs;

```

```

        cdFromDB.track[6].lengthMs = cd2id.track[6].endMs -
        cd2id.track[6].beginMs;
        cdFromDB.numberTracks = 7;

        for(i=1;i<cdFromDB.numberTracks;i++) {
            printf ("CD #2: Track = %d    length inminutes = %f\n",
i, (float)cdFromDB.track[i].lengthMs/60000.0 );
        }

        CreateUniqueId( &cd2UID, &cd2id );

        printf( "Unique ID for CD #1 = %d%d\n", cd2UID.id[0],
cd2UID.id[1] );

        CreateUniqueId( &cdFromDBUID, &cdFromDB );
        printf( "Unique ID for CD #2 = %d%d\n", cdFromDBUID.id[0],
cdFromDBUID.id[1] );

        matchtest = (cd2UID.id[0] == cdFromDBUID.id[0]) &&
(cd2UID.id[1] == cdFromDBUID.id[1]);

        printf ("The cd's match if result is non zero
matchresult=%d",matchtest);

        printf ("\n");

        printf ("\n");
        printf ("Test #2 will compare two cd that are nearly the
same\nexcept they have diffent # of tracks \n");

        // put in some test values for the cd track lengths
        // since thes are in ms, its basically 60000 = 1 minute
        cd2id.track[0].beginMs = 0;
        cd2id.track[1].beginMs = 100001;
        cd2id.track[2].beginMs = 231001;
        cd2id.track[3].beginMs = 345001;
        cd2id.track[4].beginMs = 435001;
        cd2id.track[5].beginMs = 460001;
        cd2id.track[6].beginMs = 590001;
        cd2id.track[0].endMs = 100000;
        cd2id.track[1].endMs = 231000;
        cd2id.track[2].endMs = 345000;
        cd2id.track[3].endMs = 435000;
        cd2id.track[4].endMs = 460000;
        cd2id.track[5].endMs = 590000;

```

```

        cd2id.track[6].endMs = 690000;

        cd2id.track[0].lengthMs = cd2id.track[0].endMs -
cd2id.track[0].beginMs;
        cd2id.track[1].lengthMs = cd2id.track[1].endMs -
cd2id.track[1].beginMs;
        cd2id.track[2].lengthMs = cd2id.track[2].endMs -
cd2id.track[2].beginMs;
        cd2id.track[3].lengthMs = cd2id.track[3].endMs -
cd2id.track[3].beginMs;
        cd2id.track[4].lengthMs = cd2id.track[4].endMs -
cd2id.track[4].beginMs;
        cd2id.track[5].lengthMs = cd2id.track[5].endMs -
cd2id.track[5].beginMs;
        cd2id.track[6].lengthMs = cd2id.track[6].endMs -
cd2id.track[6].beginMs;
        cd2id.numberTracks = 7;

        for(i=1;i<cd2id.numberTracks;i++) {
            printf ("CD #1: Track = %d    length inminutes = %f\n",
i, (float)cd2id.track[i].lengthMs/60000.0 );
        }

        printf ("\n");
        cdFromDB.track[0].beginMs = 0;
        cdFromDB.track[1].beginMs = 100001;
        cdFromDB.track[2].beginMs = 231001;
        cdFromDB.track[3].beginMs = 345001;
        cdFromDB.track[4].beginMs = 435001;
        cdFromDB.track[5].beginMs = 460001;
        cdFromDB.track[6].beginMs = 590001;

        cdFromDB.track[0].endMs = 100000;
        cdFromDB.track[1].endMs = 231000;
        cdFromDB.track[2].endMs = 345000;
        cdFromDB.track[3].endMs = 435000;
        cdFromDB.track[4].endMs = 460000;
        cdFromDB.track[5].endMs = 590000;

        cdFromDB.track[0].lengthMs = cd2id.track[0].endMs -
cd2id.track[0].beginMs;
        cdFromDB.track[1].lengthMs = cd2id.track[1].endMs -
cd2id.track[1].beginMs;
        cdFromDB.track[2].lengthMs = cd2id.track[2].endMs -
cd2id.track[2].beginMs;

```

```
        cdFromDB.track[3].lengthMs = cd2id.track[3].endMs -
cd2id.track[3].beginMs;
        cdFromDB.track[4].lengthMs = cd2id.track[4].endMs -
cd2id.track[4].beginMs;
        cdFromDB.track[5].lengthMs = cd2id.track[5].endMs -
cd2id.track[5].beginMs;
        cdFromDB.numberTracks = 6;

        for(i=1;i<cdFromDB.numberTracks;i++) {
            printf ("CD #2: Track = %d    length inminutes = %f\n",
i, (float)cdFromDB.track[i].lengthMs/60000.0 );
        }

        CreateUniqueId( &cd2UID, &cd2id );
        printf( "Unique ID for CD #1 = %d%d\n", cd2UID.id[0],
cd2UID.id[1] );

        CreateUniqueId( &cdFromDBUID, &cdFromDB );
        printf( "Unique ID for CD #2 = %d%d\n", cdFromDBUID.id[0],
cdFromDBUID.id[1] );

        matchtest = (cd2UID.id[0] == cdFromDBUID.id[0]) &&
(cd2UID.id[1] == cdFromDBUID.id[1]);

        printf ("The cd's match if result is non zero
matchresult=%d",matchtest);
        printf ("\n");
        printf ("\n");
    }
```


CLAIMS

I claim:

1. A method of synchronizing content from a first CD with complimentary content from a remote device delivered over a network, characterized by:
 - playing a first CD(30) on a local electronic
 - 5 device(10) to manifest first CD content;
 - transferring complimentary content (60) and
 - synchronization information (72) from a remote device (40) to
 - said local device over a network; and
 - controlling said first CD using said synchronization
 - 10 information to synchronize manifestation of said first CD
 - content with said complimentary content.
2. The method of claim 1 further characterized by:
 - transferring CD status information (62) from said
 - first CD to said remote device over a network; and
 - controlling manifestation of said complimentary
 - 5 content using CD status information and said synchronization
 - information.
3. The method of claim 1 wherein said first CD includes audio content.
4. The method of claim 1 wherein said first CD includes audio and video content.
5. The method of claim 1 wherein said first CD includes music.
6. The method of claim 1 further comprising:
 - analyzing said first CD;
 - computing an identification code based on said content
 - of said first CD; and

5 transferring complimentary content and synchronization information from said remote device based on said identification code.

7. The method of claim 1 wherein said remote device is a network server.

8. The method of claim 1 wherein said remote device is a second user device.

9. A device for synchronizing content from a first prerecorded media with complimentary content from a remote device delivered over a network, comprising:

 a local electronic device having a video display
5 means;
 a prerecorded content player connected to or integral with said local electronic device;
 means for transferring complimentary content and synchronization information from a remote device to said local
10 device over a network; and
 means for controlling said prerecorded content player using said synchronization information to synchronize manifestation of said first prerecorded media content with said complimentary content.

10. The device of claim 9 further comprising:
 means for transferring prerecorded media status information from said first prerecorded media to said remote device over a network; and
5 means for controlling manifestation of said complimentary content using prerecorded media status information and said synchronization information.

11. The device of claim 9 wherein said first prerecorded media is a CD.

12. The device of claim 9 wherein said first prerecorded media is a DVD.

13. The device of claim 9 wherein said first prerecorded media includes audio content.

14. The device of claim 9 wherein said first prerecorded media includes audio and video content.

15. The device of claim 9 wherein said first prerecorded media includes music.

16. The device of claim 9 further comprising:
means for analyzing said first prerecorded media;
means for computing an identification code based on
said content of said first prerecorded media; and
5 means for transferring complimentary content and
synchronization information from said remote device based on
said identification code.

17. The device of claim 9 wherein said remote device is a network server.

18. The device of claim 9 wherein said remote device is a second user device

1/6

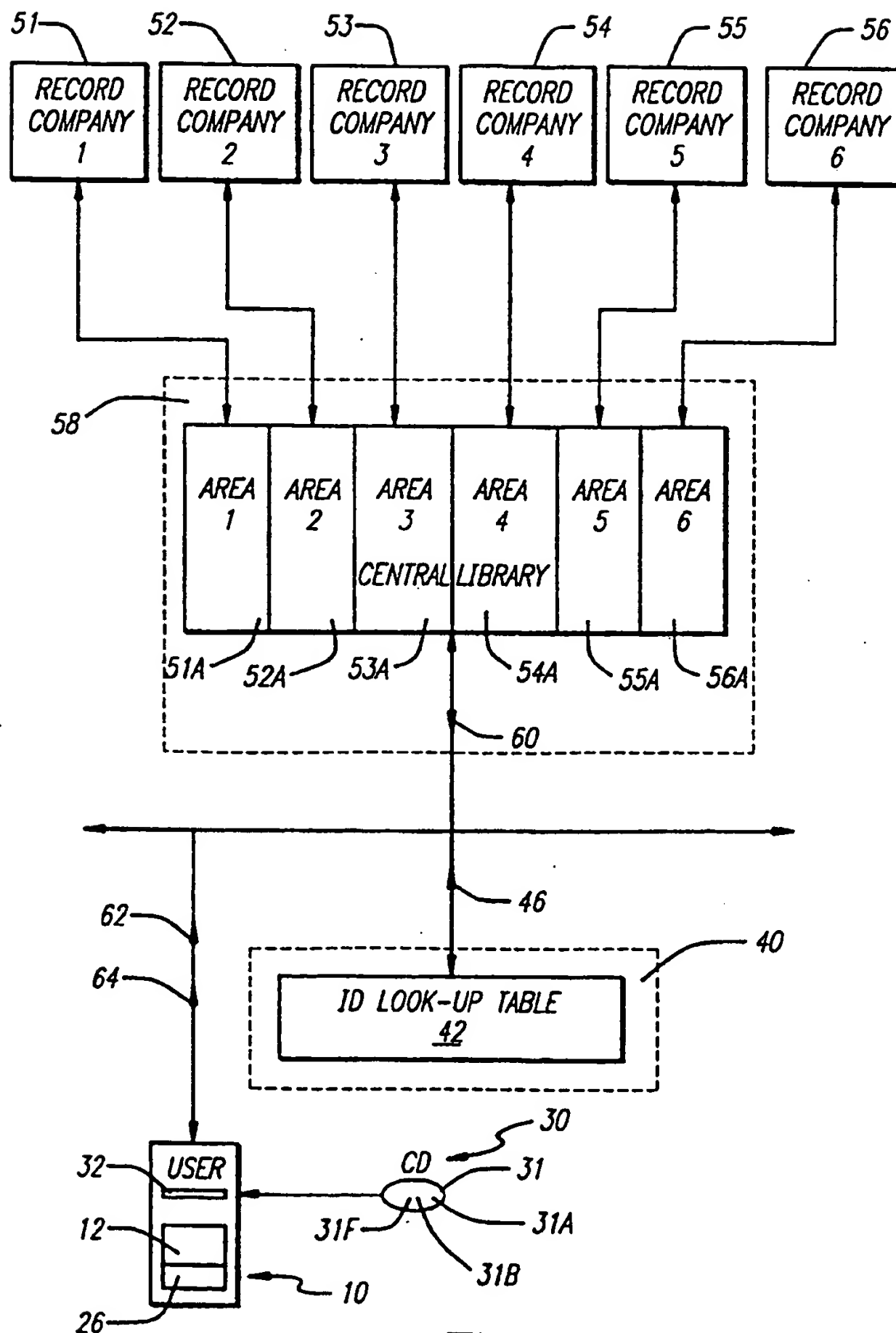


FIG. 1

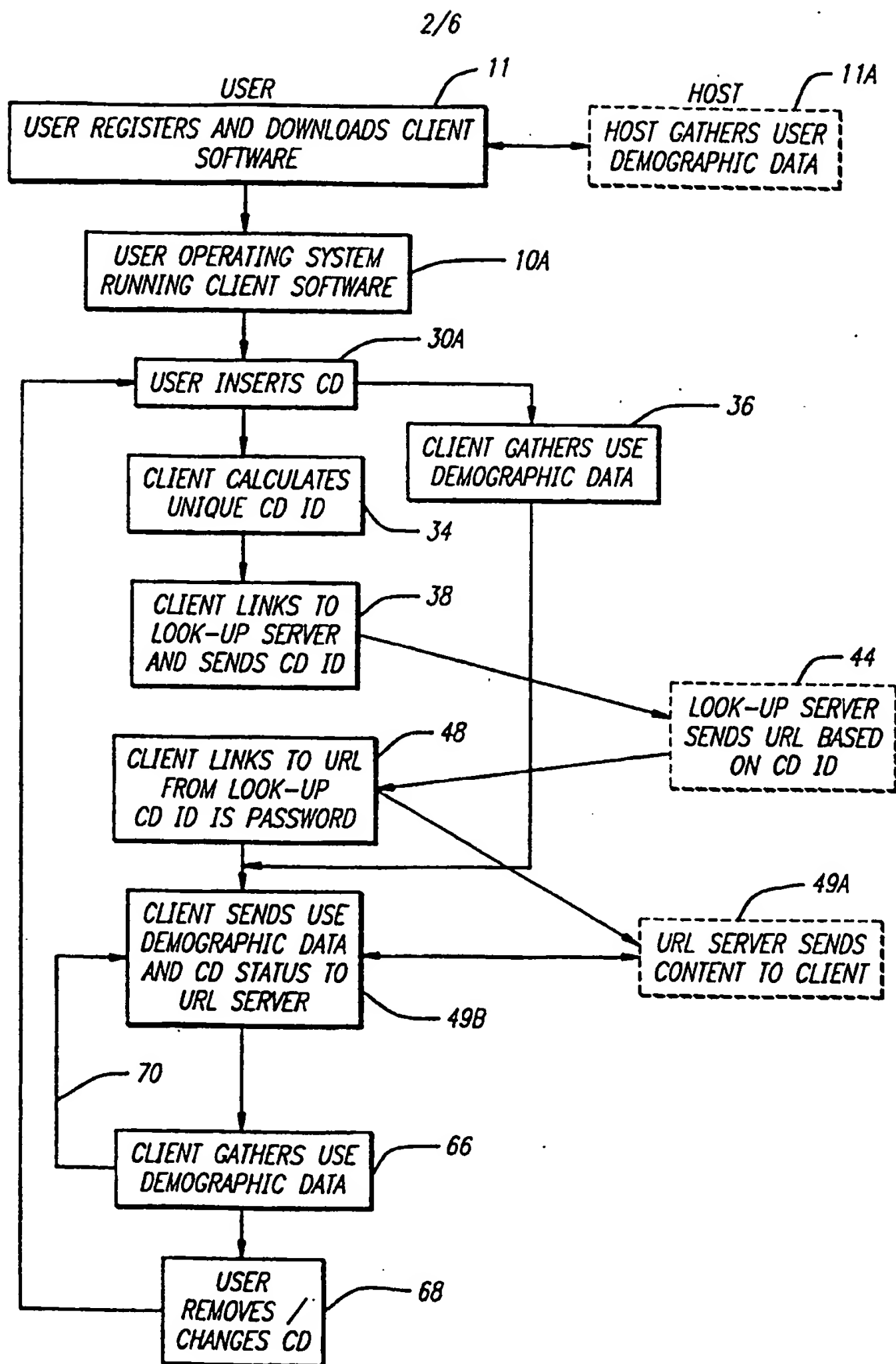


FIG. 2

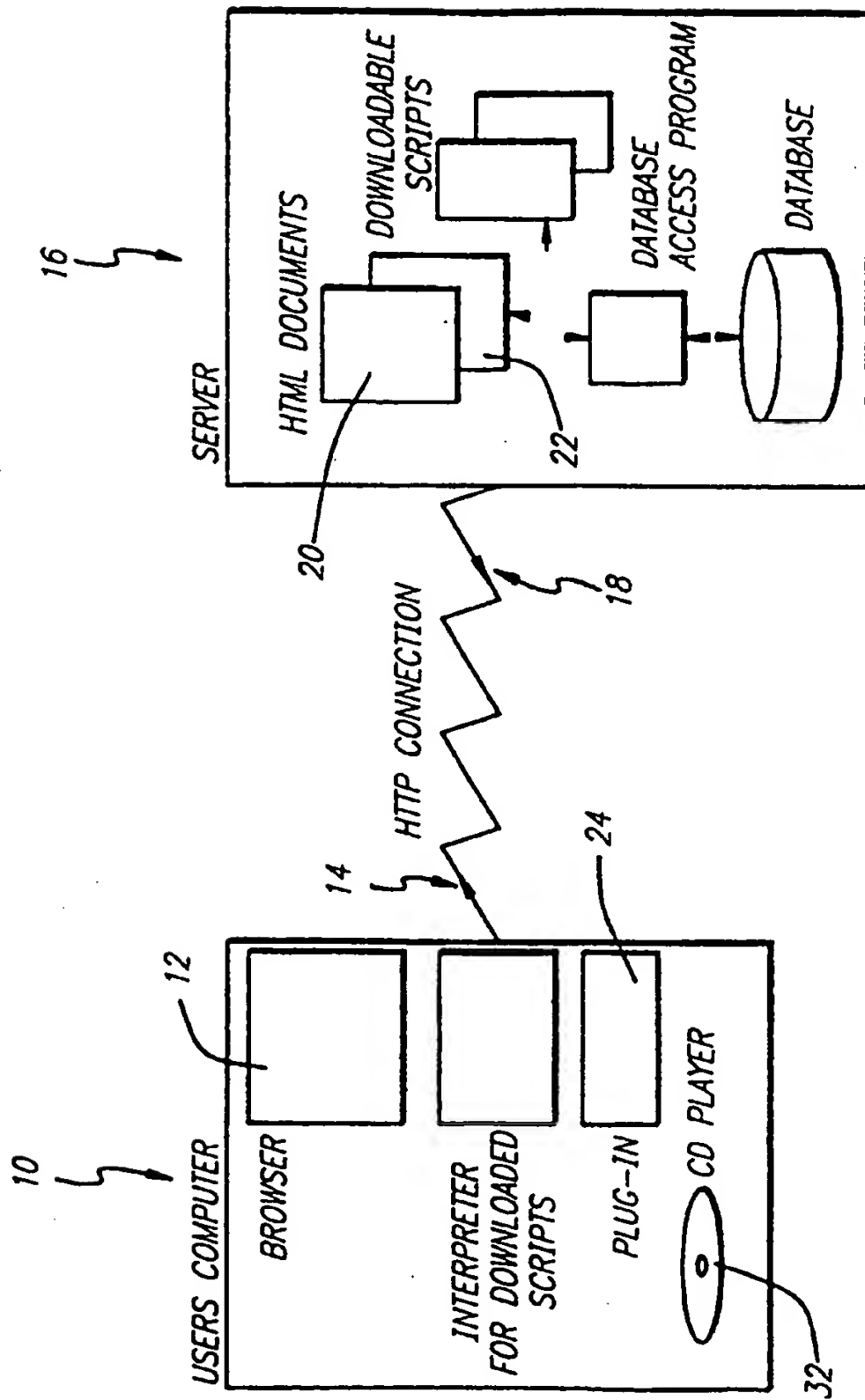


FIG. 3

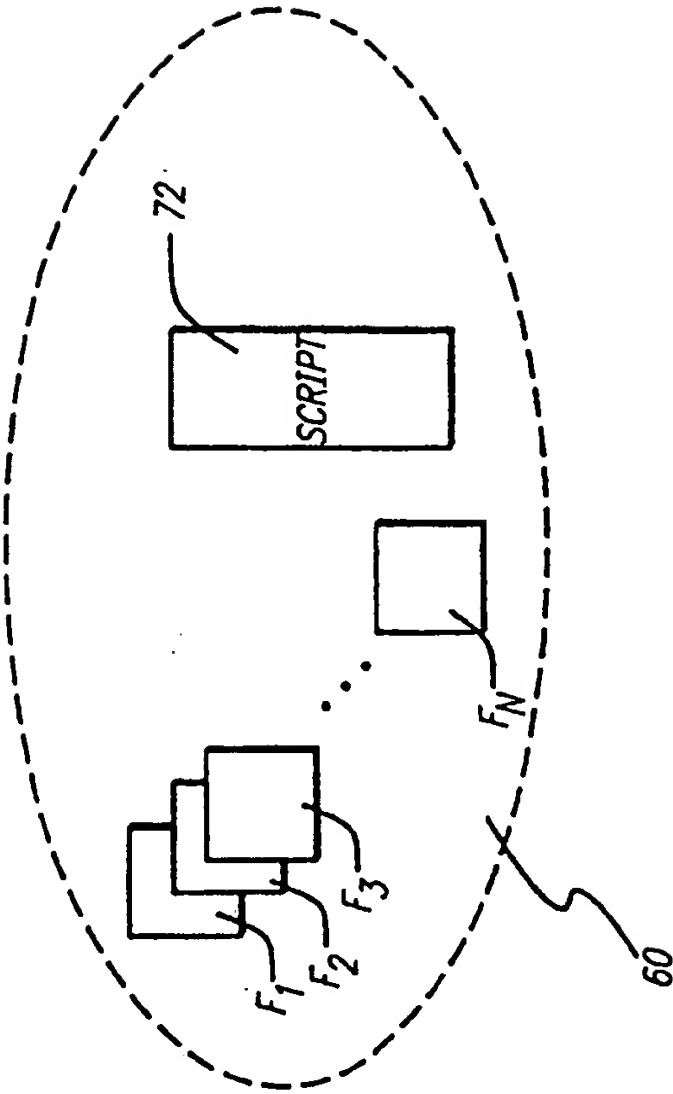


FIG. 4

5/6

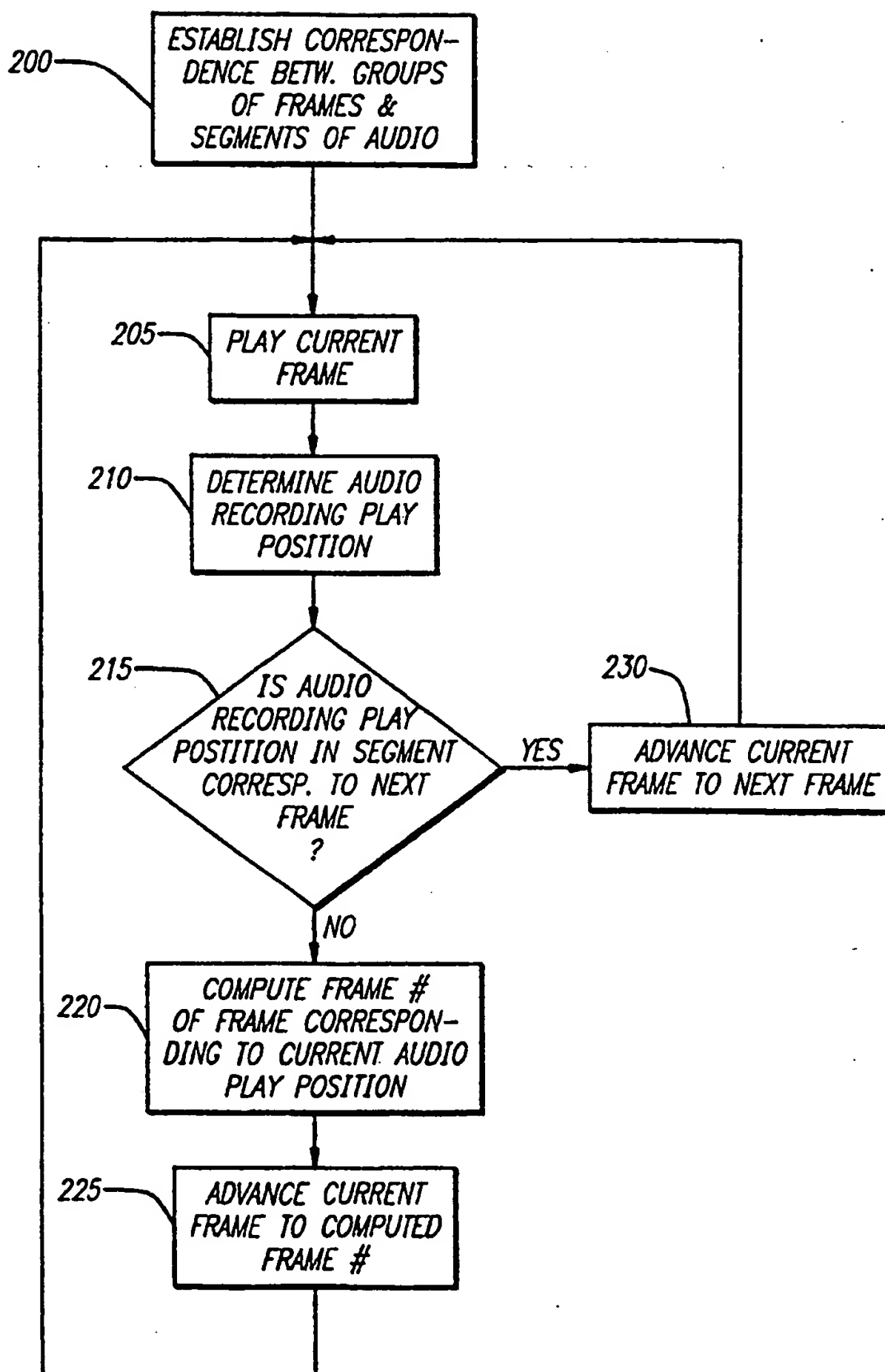


FIG. 5

6/6

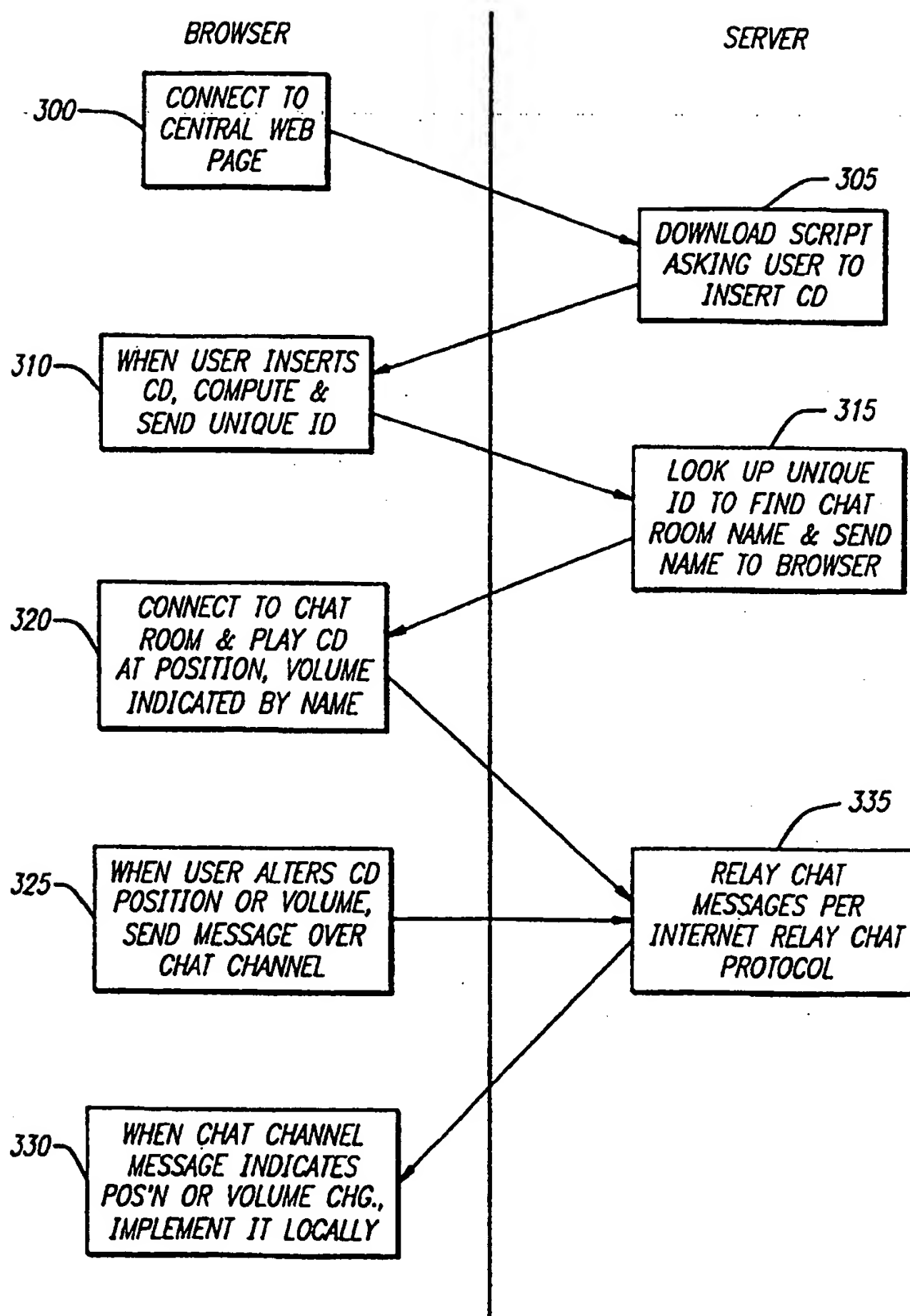


FIG. 6